CLAIM AMENDMENTS

Claim Amendment Summary

Claims pending

- Before this Amendment: Claims 1-28, 34-42, and 45-46.
- After this Amendment: Claims 1-28, 34-42, and 45-50.

Non-Elected, Canceled, or Withdrawn claims: 29-33, 43, and 44.

Amended claims: 1, 13, 34, 40, and 45.

New claims: 46-50.

Claims:

1. (**CURRENTLY AMENDED**) A kernel emulator for non-native program modules, the kernel emulator ~~being~~ comprising software ~~comprised of computer-executable instructions that are tangibly embodied on one or more computer-readable media~~ and the kernel emulator comprising:

an interceptor configured to intercept kernel calls from non-native program modules;

a call-converter configured to convert non-native kernel calls intercepted by the interceptor into native kernel calls.

2. (**ORIGINAL**) An emulator as recited in claim 1, wherein the call-converter comprises a translator configured to translate a non-native paradigm for passing parameters into a native paradigm for passing parameters.

3. (**ORIGINAL**) An emulator as recited in claim 1, wherein the call-converter comprises a translator configured to translate non-native CPU instructions into native CPU instructions.

4. (**ORIGINAL**) An emulator as recited in claim 1, wherein the call-converter comprises a translator configured to translate addresses from non-native length into native length.

5. (**ORIGINAL**) An emulator as recited in claim 1, wherein the call-converter comprises an argument-converter configured to convert non-native argument format into native argument format.

6. (**ORIGINAL**) An emulator as recited in claim 1, wherein the call-converter comprises a translator configured to translate words from non-native word size into native word size.

7. (**ORIGINAL**) An emulator as recited in claim 1 further comprising a memory constrainer configured to limit addressable memory to a range addressable by non-native program modules.

8. (**ORIGINAL**) An emulator as recited in claim 1 further comprising a shared-memory manager configured to manage memory space that is accessible to both native and non-native program modules.

9. (**ORIGINAL**) An emulator as recited in claim 1 further comprising a shared-memory manager configured to synchronize a native shared data structure with a non-native shared data structure.

10. (**PREVIOUSLY PRESENTED**) An emulator as recited in claim 1 further comprising a shared-memory manager configured to manage memory space that is accessible to both native and non-native program modules, wherein the shared-memory manager maps versions of process shared data structures (process SDSs) and versions of thread shared data structures (thread SDSs) between native and non-native program modules.

11. (**ORIGINAL**) An operating system on a computer-readable medium, comprising:

a native kernel configured to receive calls from native program modules;

a kernel emulator as recited in claim 1 configured to receive calls from non-native program modules.

12. (**ORIGINAL**) An operating system on a computer-readable medium, comprising:

a native kernel configured to receive calls from native APIs;

a kernel emulator as recited in claim 1 configured to receive calls from non-native APIs.

13. (**CURRENTLY AMENDED**) A method of emulating a kernel for non-native program modules, the method comprising:

intercepting kernel calls from non-native program modules, the kernel calls calling a kernel emulator ~~being~~ comprising software ~~comprised of computer-executable instructions that are tangibly embodied on one or more computer-readable media~~;

converting the intercepted non-native kernel calls into native kernel calls.

14. (**ORIGINAL**) A method as recited in claim 13, wherein the converting step comprises translating a non-native paradigm for passing parameters into a native paradigm for passing parameters.

15. (**ORIGINAL**) A method as recited in claim 13, wherein the converting step comprises translating non-native CPU instructions into native CPU instructions.

16. (**ORIGNAL**) A method as recited in claim 13, wherein the converting step comprises translating addresses from non-native length into native length.

17. (**ORIGINAL**) A method as recited in claim 13, wherein the converting step comprises translating words from non-native word size into native word size.

18. (**ORIGINAL**) A method as recited in claim 13 further comprising limiting addressable memory to a range addressable by non-native program modules.

19. (**ORIGINAL**) A method as recited in claim 13 further comprising synchronizing a native shared data structure with a non-native shared data structure.

20. (**ORIGINAL**) A method as recited in claim 13 further comprising mapping versions of process shared data structures (SDSs) between native and non-native program modules.

21. (**ORIGINAL**) A method as recited in claim 20, wherein a process SDS of a native program module includes a pointer to a process SDS of a non-native program module.

22. (**ORIGINAL**) A method as recited in claim 20, wherein a process SDS of a non-native program module includes a pointer to a process SDS of a native program module.

23. (**ORIGINAL**) A method as recited in claim 13 further comprising mapping versions of thread shared data structures (SDSs) data structure between native and non-native program modules.

24. (**ORIGINAL**) A method as recited in claim 23, wherein a thread SDS of a native program module includes a pointer to a thread SDS of a non-native program module.

25. (**ORIGINAL**) A method as recited in claim 23, wherein a thread SDS of a non-native program module includes a pointer to a thread SDS of a native program module.

26. (**ORIGINAL**) A computer comprising one or more computer-readable media having computer-executable instructions that, when executed by the computer, perform the method as recited in claim 13.

27. (**ORIGINAL**) A computer-readable medium having computer-executable instructions that, when executed by a computer, performs the method as recited in claim 13.

28. (**ORIGINAL**) An operating system embodied on a computer-readable medium having computer-executable instructions that, when executed by a computer, performs the method as recited in claim 13.

29. (**CANCELED**) A method comprising:

determining whether an initiating program module is a native or non-native;

if the initiating program is non-native:

limiting available memory to a range that is addressable by the non-native program module, that range of addressable memory being less that the available memory;

establishing non-native a version of a shared memory data structure that may be synchronized with a native version of the same shared memory data structure.

30. (**CANCELED**) A method as recited in claim 29 further comprising:

intercepting kernel calls from the non-native program module;

converting the intercepted non-native kernel calls into native kernel calls.

31. (**CANCELED**) A method as recited in claim 29 further comprising emulating a non-native kernel for which kernel calls from the non-native program module are intended.

32. (**CANCELED**) A computer comprising one or more computer-readable media having computer-executable instructions that, when executed by the computer, perform the method as recited in claim 29.

33. (**CANCELED**) A computer-readable medium having computer-executable instructions that, when executed by a computer, performs the method as recited in claim 29.

34. (**CURRENTLY AMENDED**) A method comprising emulating a non-native kernel for a native computing platform so that kernel calls from non-native applications are translated into calls to a native kernel, the native kernel emulator ~~being~~ comprising software ~~comprised of computer-executable instructions that are tangibly embodied on one or more computer-readable media~~.

35. (**ORIGINAL**) A method as recited in claim 34, wherein the emulating step comprises:

translating non-native CPU instructions into native CPU instructions;

translating addresses from non-native length into native length;

limiting addressable memory to a range addressable by non-native program modules.

36. (**ORIGINAL**) A method as recited in claim 35, wherein the emulating step further comprises translating a non-native paradigm for passing parameters into a native paradigm for passing parameters.

37. (**ORIGINAL**) A method as recited in claim 34, wherein the converting step further comprises translating words from non-native word size into native word size.

38. (**ORIGINAL**) A computer comprising one or more computer-readable media having computer-executable instructions that, when executed by the computer, perform the method as recited in claim 34.

39. (**ORIGINAL**) A computer-readable medium having computer-executable instructions that, when executed by a computer, performs the method as recited in claim 34.

40. (**CURRENTLY AMENDED**) A kernel emulator configured to emulate a non-native kernel for a native computing platform so that kernel calls from non-native applications are translated into calls to a native kernel, the kernel emulator ~~being~~ comprising software ~~comprised of computer-executable instructions that are tangibly embodied on one or more computer-readable media~~.

41. (**ORIGINAL**) An emulator as recited in claim 40, wherein the emulator comprises:

an instruction-translator configured to translate non-native CPU instructions into native CPU instructions;

an address-translator configured to translate addresses from non-native length into native length;

an memory constrainer configured to limit addressable memory to a range addressable by non-native program modules.

42. (**PREVIOUSLY PRESENTED**) An operating system on a computer-readable medium, comprising:

a native kernel configured to receive calls from native program modules;

a kernel emulator as recited in claim 40 configured to receive calls from non-native program modules.

43. (**CANCELED**)

44. (**CANCELED**)

45. (**CURRENTLY AMENDED**) A kernel emulator for non-native program modules, the kernel emulator ~~being~~ comprising software ~~comprised of computer-executable instructions that are tangibly embodied on one or more computer-readable media~~ and the kernel emulator comprising:

an interceptor configured to intercept kernel calls from non-native program modules;

a call-converter configured to convert non-native kernel calls intercepted by the interceptor into native kernel calls, wherein the call-converter comprises:

an instruction-translator configured to translate non-native CPU instructions into native CPU instructions;

an address-translator configured to translate addresses from non-native length into native length.

46. (**ORIGINAL**) An operating system on a computer-readable medium, comprising:

a native kernel configured to receive calls from native program modules;

a kernel emulator as recited in claim 45 configured to receive calls from non-native program modules.

--- NEW ---

47. (**NEW**) A kernel emulator for non-native program modules, the kernel emulator comprising software and the kernel emulator comprising:

an interceptor configured to intercept kernel calls from non-native program modules;

a call-converter configured to convert non-native kernel calls intercepted by the interceptor into native kernel calls;

a call-deliverer configured to deliver the native kernel calls—which are the results of conversion by the call-converter—to a native kernel.

48. (**NEW**) A method comprising emulating a non-native kernel for a native computing platform so that kernel calls from non-native applications are translated into calls to a native kernel, the native kernel emulator.

49. (**NEW**) A kernel emulator comprising software and configured to emulate a non-native kernel for a native computing platform so that kernel calls from non-native applications are translated into calls to a native kernel thereby enabling applications dependent on non-native kernel calls to execute independent of the non-native kernel.

50. (**NEW**) A method of emulating a kernel for non-native program modules thereby enabling applications dependent on non-native kernel calls to execute independent of the kernel for non-native program modules, the method being implemented at least partially by a computer, the method comprising:

intercepting kernel calls from non-native program modules, the kernel calls calling a kernel emulator comprising software;

converting the intercepted non-native kernel calls into native kernel calls;

outputting the native kernel calls.